CLAIMS

1. Process for assigning tasks in a multiprocessor digital data processing system with a preemptive operating system, comprising a given number of processors capable of processing said tasks in parallel, characterized in that it comprises at least one preliminary phase during which said processors (20*a*-21*a*, 20*b*-22*b*, 20*c*) are divided into groups (G*a*, G*b*, G*c*), each group comprising predetermined numbers of processors, in that each of said processor groups is associated with an elementary queue (5*a*, 5*b*, 5*c*) storing a predetermined number of tasks to be processed in a given order of priority, and in that each of the tasks is associated with one of the processors associated with this elementary queue (5*a*, 5*b*, 5*c*).

2. Process according to claim 1, characterized in that said groups each comprise an identical number of processors (200-203, 210-213).

3. Process according to claim 1 or 2, characterized in that it comprises an additional preliminary phase consisting of generating a series of tests and measurements for determining the number of processors in each group and the number of groups that make it possible to achieve the best performance of said system.

4. Process according to any of claims 1 through 3, characterized in that, the architecture of said system being of the non-uniform memory access type known as "NUMA," and the system (1) being constituted by a predetermined number of modules (*M*0, *M*1) linked to one another, each comprising a given number of processors (200-203, 210-213) and storage means, each of said modules (*M*0, *M*1) constitutes one of said groups, each module being associated with one of said elementary queues.

5. Process according to any of claims 1 through 4, characterized in that each of said processors is associated with a first data structure that identifies it, in that said first data structure comprises at least one first set of pointers (*p*200 through *p*203) associating it with one of said

elementary queues (5*a*, 5*b*), in that each of said elementary queues (5*a*, 5*b*) is associated with a second data structure, in that said second data structure comprises at least one second set of pointers (*pp*5*a*, *pp*5*b*) associating it with one of said processor groups (200-201, 202-203), in that all of the tasks to be processed (*T*1 through *T*10) in said system (1) being stored in a table (4), each of said second data structures of the elementary queues (5*a*, 5*b*) also comprises a third set of pointers (*pT*1, *pT*5, *pT*10), each associating elementary queues (5*a*, 5*b*) with one of said tasks (*T*1 through *T*10) stored in the table (4) or with a series of concatenated tasks, and in that each of said tasks (*T*1 through *T*10) of the table (4) is associated with a third data structure that comprises a fourth set of pointers (*p*5*a*1 through *p*5*a*4, *p*5*b*1 through *p*5*b*10) associating it with one of said elementary queues (5*a*, 5*b*).

6. Process according to any of claims 1 through 5, characterized in that it comprises at least one additional phase consisting of distributing said tasks among said elementary queues (5*a*, 5*b*) by searching, when a new task to be processed (T*z*) is created, for the queue with the lightest load (5*y*) among all of said elementary queues (5*a*, 5*x*, 5*y*, 5*p*) of said system (1) and of assigning said new task to this elementary queue so as to balance the global load of this system (1) among said elementary queues (5*a*, 5*x*, 5*y*, 5*p*).

7. Process according to claim 6, characterized in that said distribution is performed by determining a so-called composite load parameter associated with each of said elementary queues (5*a*, 5*x*, 5*y*, 5*p*) and in that, each processor (2*a*, 2*x*, 2*y*, 2*p*) being associated with storage means (Mem*a*, Mem*x*, Mem*y*, Mem*p*), said composite load parameter is calculated as being the sum of the load of a processor or a processor group associated with said elementary queue and the load of the storage means associated with this processor or processor group.

8. Process according to claim 6, characterized in that it comprises a preliminary step consisting of checking whether said task (T*z*) is linked to one of said elementary queues (5*a*, 5*x*, 5*y*, 5*p*), and when said test is positive, of assigning said new task to this elementary queue.

~~9. Process according to any of claims 1 through 5, characterized in that it comprises at least one additional phase consisting, when one of said elementary queues (5*q*) associated with one of said processor groups (2*q*) is empty of executable tasks, of searching for a so-called remote elementary queue (5*y*) that is not empty, and of selecting in this elementary queue (5*y*) a task executable by one of said processors (2*q*) of said processor group associated with the empty elementary queue (5*q*) and of transmitting it to this processor (2*q*) to be processed by it, so as to globally balance the processing of said tasks in said system (1).~~

~~10. Process according to claim 9, characterized in that said non-empty elementary queue (5*y*) must have a predetermined minimal occupation threshold.~~

~~11. Process according to claim 10, characterized in that furthermore, the tasks being stored in decreasing order of priority, a predetermined number of tasks is skipped before scanning the other tasks of said non-empty elementary queue (5*y*) in order to search for an executable task and have it processed by one of said processors (2*q*) of said processor group associated with the empty elementary queue (5*q*).~~

~~12. Process according to claim 11, characterized in that said number of skipped tasks and the maximum number of scanned tasks among all of those stored in said non-empty elementary queue (5*q*) are variable over time and are determined by a self-adapting process from the number of tasks that are or are not found during said scans and from the position of these tasks, sequenced in order of priority, in said non-empty elementary queue (5*q*).~~

~~13. Process according to any of claims 9 through 12, characterized in that said selected task is that associated with a minimal value of a so-called cost parameter, which measures the global performance degradation of said system (1) due to the processing of said selected task in said non-empty remote elementary queue (5*q*) by one of said processors of said processor group associated with the empty elementary queue (2*q*).~~

14. Process according to any of claims 1 through 5, characterized in that it comprises at least one additional phase comprising at least one step for periodically measuring for a balanced distribution of said tasks in said elementary queues (5*a*, 5*x*, 5*y*, 5*p*) and, when an unbalanced state of said system (1) is determined, a step for selectively moving tasks from at least one elementary queue with a heavier load (5*x*) to an elementary queue with a lighter load (5*y*).

15. Process according to claim 14, characterized in that when said imbalance is below a certain threshold, no moving of tasks is performed.

16. Process according to claim 14 or 15, characterized in that, all or some of said tasks belonging to multitask processes, each multitask process requiring a given memory size and workload, it comprises a step for measuring said workloads and said memory sizes, in that it comprises the selection of the process requiring the greatest workload and the smallest memory size, and in that all the tasks of said selected process are moved to the elementary queue with the lightest load (5*y*).

17. Process according to claim 16, characterized in that it comprises a preliminary step consisting of checking whether all of the tasks of said multitask process that must be moved belong to the elementary queue set with the heaviest load (5*x*) and whether any task is linked to any of said groups.

18. Process according to any of claims 1 through 17, characterized in that said operating system is of the "UNIX" (registered trademark) type.

19. Architecture for a multiprocessor digital data processing system comprising a given number of processors for implementing the process for assigning tasks to be processed to said processors according to any of claims 1 through 18, characterized in that said processors (20*a*-21*a*, 20*b*-22*b*, 20*c*) are divided into groups (G*a*, G*b*, G*c*), and in that an elementary queue

~~(5*a*, 5*b*, 5*c*) associated with each of the groups (G*a*, G*b*, G*c*) is provided, each of said elementary queues (5*a*, 5*b*, 5*c*) storing a predetermined number of tasks to be processed in a given order of priority, so that each of the tasks of each of said elementary queues (5*a*, 5*b*, 5*c*) is associated with one of the processors of this elementary queue (20*a*-21*a*, 20*b*-22*b*, 20*c*).~~

~~20. Architecture according to claim 19, characterized in that it also comprises means (6) for determining the load of said elementary queues (5*a*, 5*x*, 5*y*, 5*p*) and for assigning a new task created in said system to the elementary queue with the lightest load (5*y*).~~

~~21. Architecture according to claim 19, characterized in that it also comprises, when one (5*q*) of said elementary queues (5*a*, 5*x*, 5*y*, 5*p*) associated with one of said processors (2*q*) is empty, means (7) for finding a non-empty, so-called remote elementary queue (5*y*), and for finding an executable task in this elementary queue (5*y*), and assigning it to said processor (2*q*) for processing.~~

~~22. Architecture according to claim 19, characterized in that it also comprises means (8) for detecting an imbalance between elementary queues (5*a*, 5*x*, 5*y*, 5*p*), and when such an imbalance is detected, for determining the elementary queue with the heaviest load (5*x*) and the elementary queue with the lightest load (5*y*), and for moving tasks from the elementary queue with the heaviest load (5*x*) to the elementary queue with the lightest load (5*y*).~~

~~23. Architecture according to any of claims 19 through 22, characterized in that, being of the nonuniform memory access type known as "NUMA," composed of modules (*M*0, *M*1) linked to one another, each comprising a given number of processors (200-203, 210-213) and storage means, each of said modules (*M*0, *M*1) constitutes one of said groups, each module (*M*0, *M*1) being associated with one of said elementary queues.~~